AMENDMENTS TO THE SPECIFICATION

In the Specification:

Please replace the paragraph beginning on page 1, line 24, with the following amended paragraph:

As noted above, the most common algorithm that is implemented is the C-LOOK algorithm which orders requests in order that the disk head sweeps in one direction across the disk performing Input/Output operations (I/Os), then returns to the other side of the disk to perform additional I/Os. This is often referred to as logical block address ordering or LBA ordering. Unfortunately, traditional disk scheduling algorithms trade-off higher throughput to the disk, for longer access times (or latencies), over simple first-in-first-out (FIFO) queuing. This provides better throughput, but at the cost of potentially and dramatically higher latencies for I/O requests. For traditional operating systems, this trade-off is well worth the cost. For newer systems that need to support audio-visual streaming applications, in one example, merely trading latency for throughput is often not acceptable.

Please replace the paragraph beginning on page 2, line 26, with the following amended paragraph:

The present invention relates to systems and methods that facilitate dynamic scheduling of data requests to a storage media in an efficient[[,]] and timely manner. The data requests are processed as Input/Output (I/O) reads from, and writes to, the storage media such as a disk drive or other type media. Scheduling is achieved in an architecture that reorders a fixed number of requests per scan of the disk drive, thus providing a fixed upper bound on the latency of the request wherein each scan of the disk drive is referred to as a *round*. Since the requests are reordered into an efficient pattern for the disk drive

to execute, high throughput to the disk drive is also achieved. Thus, a disk scheduling system is provided that includes at least one scheduling component. The scheduling component employs a predetermined number of requests within a round to provide a particular latency guarantee for the requests while maintaining high throughput level in connection with disk updates.

Please replace the paragraph beginning on page 5, line 25, with the following amended paragraph:

The categorizer 130 transfers requests into the queues 140 that are processed by a scheduler 160, having a fixed or predetermined size queue 170. The scheduler 160 transfers data to/from the queue 170 in order to read data from and/or write data to the storage media 120 in the form of one or more rounds of data [[180]] 150 that are directed to the media in accordance with determined slots of time that bound the overall latency and throughput of the system. By transferring rounds [[180]] 150 of data requests having predetermined size in terms of numbers of requests per round, the present invention dynamically balances the overall latency to access the storage media 120 in view of a desired data throughput level to/from the media.

Please replace the paragraph beginning on page 6, line 5, with the following amended paragraph:

As noted above, the data requests 110 are processed as I/O ~~reads/writes~~ reads from and/or writes to the storage media 120 which can include disk drives (*e.g.,* hard disk, floppy disk) or other type media (*e.g.,* CD, any type of memory that is scheduled to be accessed from a computer). Scheduling is generally provided in a multi-tiered architecture that bounds the latency of I/O requests by servicing a predetermined number, subset, or set of requests per round of requests 150 while maintaining desired throughput to the storage media 120. Also, a determined amount of I/O bandwidth can be dynamically reserved by the categorizer 130 and/or the scheduler 160 in order to allocate I/O scheduling for selected tasks. It is noted that the components illustrated in the system 100 can be isolated and ~~communicate~~ communicated on local and/or remote computer systems, and/or can be combined in various forms to perform the functionality described herein (*e.g.,* ~~pre-~~categorizer and scheduler combined to form a scheduling function).

Please replace the paragraph beginning on page 8, line 1, with the following amended paragraph:

The sweep queue 240 is employed to queue requests that will be scheduled to the physical disk 250. Requests from the class-specific queues are moved to the sweep queue as it empties. The elements in the sweep queue are ordered in logical-block-address (LBA) ordering (other orderings possible), wherein the sweep queue performs a C-LOOK algorithm (or other similar type) when scheduling requests to the drive 250. Additionally, the system 200 fixes the size of the sweep queue 240, in order to calculate *a priori* the length of time required to schedule the entire sweep queue. Moreover, the system 200 can reserve entries in the sweep queue 240 to facilitate availability of I/O bandwidth when necessary.

Please replace the paragraph beginning on page 8, line 10, with the following amended paragraph:

Fig. 3 illustrates a system 400 that can be employed to perform an implementation of the components and process described above. One or more requests 404 for I/O disk access are received by an admission controller 410, wherein the request is generally received from a component or application executable on a computer system. If the request is a periodic request, a deadline component 420 associated with the admission controller 410 assigns a deadline or timeframe in which the request is to be completed. Other type requests such as non-periodic requests are passed from the admission controller 410 to an aperiodic queue [[430]] 440 associated with class-specifier component. The periodic requests that were tagged with deadline specifications are processed within a periodic queue [[440]] 430 within the class specifier-component. The class-specifier sorts requests based upon a desired policy associated with a selected class of requests. As noted above, the periodic queue [[440]] 430 can be arranged according to an EDF ordering, whereas the aperiodic queue [[430]] 440 can be arranged according to a FIFO ordering. When a current sweep round has completed, a sweep queue (not shown) is filled from the queues 430 and 440. The sweep queue is then scheduled to a disk [[450]] 460 (or disks) in C-Look order *via* a C-Look component [[460]] 450.

Please replace the paragraph beginning on page 10, line 11, with the following amended paragraph:

It is noted that that the above algorithm has a well-defined upper bound on the latency to complete a specified number of requests. For *n* requests, *n* being an integer, the upper-bound on latency is as follows:

$$service(n) = n\left(time\_seek\left(\frac{Cylinders}{N}\right) + time_{transfer} + time_{rotation} + time_{controller}\right) + time_{sweep}.$$

Please replace the paragraph beginning on page 11, line 16, with the following amended paragraph:

With reference to Fig.6, an exemplary environment [[710]] 77 for implementing various aspects of the invention includes a computer 712. The computer 712 includes a processing unit 714, a system memory 716, and a system bus 718. The system bus 718 couples system components including, but not limited to, the system memory 716 to the processing unit 714. The processing unit 714 can be any of various available processors. Dual microprocessors and other multiprocessor architectures also can be employed as the processing unit 714.

Please replace the paragraph beginning on page 11, line 23, with the following amended paragraph:

The system bus 718 can be any of several types of bus structure(s) including the memory bus or memory controller, a peripheral bus or external bus, and/or a local bus using any variety of available bus architectures including, but not limited to, 16-bit bus, Industrial Standard Architecture (ISA), Micro-Channel Architecture [[(MSA)]] (MCA), Extended ISA (EISA), Intelligent Drive Electronics (IDE), VESA Local Bus (VLB), Peripheral Component Interconnect (PCI), Universal Serial Bus (USB), Advanced Graphics Port (AGP), Personal Computer Memory Card International Association bus (PCMCIA), and Small Computer Systems Interface (SCSI).

Please replace the paragraph beginning on page 12, line 25, with the following amended paragraph:

It is to be appreciated that Fig. 6 describes software that acts as an intermediary between users and the basic computer resources described in suitable operating environment [[710]] 77. Such software includes an operating system 728. Operating system 728, which can be stored on disk storage 724, acts to control and allocate resources of the computer system 712. System applications 730 take advantage of the management of resources by operating system 728 through program modules 732 and program data 734 stored either in system memory 716 or on disk storage 724. It is to be appreciated that the present invention can be implemented with various operating systems or combinations of operating systems.

Please replace the paragraph beginning on page 14, line 7, with the following amended paragraph:

Communication connection(s) 750 refers to the hardware/software employed to connect the network interface 748 to the bus 718. While communication connection 750 is shown for illustrative clarity inside computer 712, it can also be external to computer 712. The hardware/software necessary for connection to the network interface 748 includes, for exemplary purposes only, internal and external technologies such as[[,]] modems, including regular telephone grade modems, cable modems, [[and]] DSL modems, ISDN adapters, and Ethernet cards.

Please replace the paragraph beginning on page 14, line 14, with the following amended paragraph:

Fig. 7 is a schematic block diagram of a sample-computing environment 800 with which the present invention can interact. The system 800 includes one or more ~~client(s)~~ clients 810. The client(s) 810 can be hardware and/or software (*e.g.,* threads, processes, computing devices). The system 800 also includes one or more ~~server(s)~~ servers 830. The server(s) 830 can also be hardware and/or software (*e.g.,* threads, processes, computing devices). The ~~servers~~ server(s) 830 can house threads to perform transformations by employing the present invention, for example. One possible communication between a client 810 and a server 830 may be in the form of a data packet adapted to be transmitted between two or more computer processes. The system 800 includes a communication framework 850 that can be employed to facilitate communications between the client(s) 810 and the server(s) 830. The client(s) 810 ~~are~~ is operably connected to one or more client data ~~store(s)~~ stores 860 that can be employed to store information local to the client(s) 810. Similarly, the server(s) 830 ~~are~~ is operably connected to one or more server data ~~store(s)~~ stores 840 that can be employed to store information local to the ~~servers~~ server(s) 830.